VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Spamové filtry založené na pravidlech

Spam Filters based on Rules

2013

Martin Popelář

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Popelář**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Spamové filtry založené na pravidlech
Spam Filters based on Rules

Zásady pro vypracování:

Vytvořte práci, v které budou popsány tři vybrané spamové filtry založené na pravidlech. Práce bude obsahovat popis vybraných algoritmů a jejich srovnání.
Práce bude obsahovat tyto části:
1. Přehled a popis spamových filtrů založených na pravidlech.
2. Srovnaní filtrů založených na pravidlech.
3. Analýza vybraných filtrů.
4. Popis implementace a vlastní implementace vybraných filtrů.
5. Srovnaní vybraných filtrů na základě provedených experimentů a testů.

Seznam doporučené odborné literatury:

[1] Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification Jonathan A. Zdziarski ISBN:1593270526

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Prílepok**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, zejména můj konzultant Ing. Michal Prílepok, protože bez nich by tato práce nevznikla.

Abstrakt

V této práci se budeme zabývat třemi vybranými spam filtry založených na pravidlech, a to filtry: FuzzySpamFiltr, RuBaSt, LazyApproach. U všech těchto spam filtrů si popíšeme podrobně jejich pravidla a algoritmy. Každý spam filtr otestujeme na anglické kolekci e-mailů od společnosti NIST. Následně porovnáme výsledky těchto tří filtrů mezi sebou.

Klíčová slova: e-mail, spam, spam filtr, FuzzySpamFiltr, RuBaSt, LazyApproach

Abstract

In this thesis we will discuss about three selected rule-based spam filters, it will be: FuzzySpamFilter, RuBaSt. LazyApproach. For all these spam filters we will describe in detail their rules and algorithms. For each spam filter we will run a test on collection of english e-mails from company NIST. After that we will compare the results.

Keywords: e-mail, spam, spam filter, FuzzySpamFilter, RuBaSt, LazyApproach

Seznam použitých zkratek a výrazů

spam	–	nevyžádané sdělení
ham	–	označení pošty, která není spamem
e-mail	–	elektronická pošta
spammer	–	rozesílatel spamu
C#	–	programovací jazyk C-sharp
open-source	–	počítačový software s otevřeným zdrojovým kódem. Otevřenost zde znamená jak technickou dostupnost kódu, tak legální dostupnost - licenci software.

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod do problematiky spamů

Internet nám dal nový způsob komunikace, například vytvořit e-mail, který bude teoreticky poslán tisíce kilometrů daleko. Toto nové médium komunikace nám otevřelo dveře pro masové rozesílání zpráv, které překonávají kilometry během sekund. Nicméně, svoboda této komunikace může být snadno zneužita. Během posledních pár let se spam stal fenoménem, který ohrožuje e-mailovou komunikaci.

Vytvořit přesnou definici spamu je složité, avšak každý uživatel e-mailů rychle a jednoduše spam rozezná. Světově uznávaný slovník Merriam-Webster Online Dictionary[1] definuje spam jako „unsolicited usually commercial e-mail sent to a large number of adress“, čili „nevyžádaný, většinou komerční e-mail, poslaný na velké množství adres“. Jiné než komerční záměry spamu jsou například vyjádření politického nebo náboženského názoru, který má přesvědčit zaměřené uživatele. Jiné spamy zase slouží pro nalákání na zajímavou nabídku, zatímco odkazují na zavirované stránky. Zvláštní a velice problémovým druhem spamu jsou e-maily, které odkazují na stránky uživateli dobře známé, ale ve skutečnosti jsou to stránky odlišné, i když vzhledem téměř identické s originálem. Cílem těchto spamů je z uživatelů dostat citlivé údaje (jako například číslo a heslo k bankovnímu účtu). Tato technika se nazývá phishing. I když by mohl někdo označit určité spamové e-maily za zajímavé, většina lidí se shodne na tom, že jsou spíše otravné.

Spam se stal závažným problémem, protože se v krátkém čase dostane k cíli a má nízké náklady, díky tomu si ho oblíbilo hodně společností, kterým nezáleží na pověsti a jde jim jen o zisk. Nízké náklady e-mailu jako komunikačního prostředku zaručují zisk. Pokud by spamový e-mail obsahoval například nabídku produktů určité firmy a z uživatelů, kterým tento spam dojde, by zareagovalo a nakoupilo si z nabídky bylo velice malé procento, stále to stojí za ten čas a peníze, vynaložené na zaslání hromadných e-mailů. Jak již bylo řečeno komerciální spamy jsou obvykle zasílány společnostmi, které nemají žádnou reputaci, takže nemohou ani žádnou ztratit. Kvůli technologickým překážkám je složité zasílatele spamu vystopovat, spammeři to mohou ještě ztížit tím, že zamaskují původ spamu. I když se je nakonec podaří vystopovat, je těžké podniknout proti nim právní kroky.

Podle společnosti Kaspersky Lab[2] spamu v posledním čtvrtletí (Q3 2012) sice ubylo ve srovnání s předchozím čtvrtletím o 2,8%, ale i tak poměr spamu z celé e-mailové komunikace dosahuje k 71,5%. Do určité míry může být pokles podílu spamu způsoben obchodním zpomalením v letním období. Nicméně, klesající trend ve výši spamu je také v důsledku postupného přesunu reklamních zpráv z e-mailu na jiné místa, jako jsou bannerové reklamy, sociální sítě, a kontextová reklama. Ve Q3 2012 opět spammeři prokázali svou vynalézavost v utajování spamu. Mezi takové e-maily, na které odborníci společnosti Kaspersky Lab narazili, byly zprávy údajně odeslané od poskytovatelů hostingu, bankovních systémů, sociálních sítí, on-line obchodů a různých dalších služeb. Celých 27% těchto phishingových útoků se přesunulo na sociální sítě. Více informací o vývoji spamu naleznete ve zprávě[3], kterou vydalo Kaspersky Lab.

[1] http://www.merriam-webster.com/dictionary/spam

[2] http://www.kaspersky.com/about

[3] http://www.securelist.com/en/analysis/204792251/Spam_in_Q3_2012

Hlavní problém spamu je, že firmy kvůli nim ztrácejí peníze ve formě času. Například pro malou společnost, která má 20 zaměstnanců a každý z nich obdrží 20 spamových e-mailů za den. Pokud vezmeme, že identifikace a odstranění spamu bude trvat 5 sekund, firma tak přijde o půl hodiny každý den jenom tím, že budou třídit e-maily. Spamy můžou způsobovat taky další problémy jako nevhodný obsah atp. Vzniká tak potřeba kontrolovat spamy, tím se dostáváme k problematice spam filtrů.

2 Techniky k eliminaci spamu

Existuje několik různých přístupů, jak se vypořádat se spamem. V téhle sekci se budeme zabývat pouze některými z nich, existuje jich mnohem více.

2.1 Schování e-mailové adresy

Nejjednodušším způsobem, jak se vyhnout spamu, je schování e-mailové adresy před spamery. Předávat e-mailovou adresu pouze důvěryhodným partnerům. Pro komunikaci s méně věrohodnými partnery může být použitý dočasný e-mail. Pokud je e-mail umístěný na webové stránce, mohou ho nalézt tzv. E-mail Spiders – roboti, prohledávající weby pro získání e-mailových adres. Zabránit tomu, aby takový robot našel vaši adresu, se dá snadno. Stačí napsat adresu nezvyklým způsobem, většina robotů hledá adresy ve tvaru jmeno@domena.cz, takovou e-mailovou adresu můžeme přepsat např. na jméno (at) domena (dot) cz. Většina lidí tomuto zápisu porozumí, zatímco před roboty zůstane skryta. Avšak pro vetší firmy je tato metoda nepoužitelná, protože může zmást nejen roboty, ale také případné zákazníky.

2.2 Podle vzorů, whitlelisty a blacklisty

Tohle je metoda založena na přístupu k obsahu příchozího e-mailu. Příchozí e-mail se zanalyzuje podle vzoru a vyhodnotí se jako spam nebo legitimní e-mail (dále jako ham). Tato technika se zabývá převážně jen porovnáváním řetězců. Má tzv. whitelisty – seznam povolených slov, a blacklisty – seznam zakázaných. Pokud příjde e-mail, který se shoduje se záznamy ve whitelistu, tak je označen jako ham. Avšak pokud příjde e-mail, který se shoduje se záznamy v blacklistu, je označen jako spam. Tato metoda může do určité míry zamezit spamu, ale vyžaduje časté aktualizace whitelistu a blacklistu. To je ovšem náročné na čas. Ve skutečnosti je tato metoda jen jakýmsi zjednodušením metody více sofistikované, a to filtrování založené na pravidlech.

2.3 Greylisting

Greylisting je velice zajímavá a efektivní metoda. Princip spočívá v tom, že automaticky odmítá všechny e-maily, které jsou od neznámého odesílatele. Odesílatel bude označen na určitou dobu (většinou 24 hodin) jako „grey“. Legitimní e-mail je často hned jak je to možné poslán znovu, pokud příjde e-mail od odesílatele, který je „grey“ tak je e-mail propuštěn dále a je označen jako legitimní. Tato metoda se začala používat, protože hodně spamerů vyúžívá na svou špinavou práci roboty, kteří e-mail odešlou pouze jednou a už je nezajímá, jestli dorazil. Kdyby měli řešit každý e-mail který nedorazil, bylo by to pro ně časově nevýhodné.

2.4 Filtrování založené na pravidlech

Tato metoda je velice populární, je to metoda založena na třídění obsahu, využívají ji například programy jako je SpamAssasin[4], který pro odhalení spamu však využívá i další techniky. Filtry založené na pravidlech aplikují na příchozí e-mail řadu pravidel. Pokud se najde shoda, e-mailu se přiřadí číselné ohodnocení, podle kterého se pak rozhodne, jestli je daný e-mail spam nebo ham. Pravidla jsou většinou napsány pomocí regulárních výrazů a jsou součástí software. Pravidla se musí často aktualizovat, protože spamy se také neustále vyvíjejí. Aktualizace pravidel většinou probíhá pomocí aktualizací přes internet. Výhoda filtrů založených na pravidlech je ta, že je nemusíte „cvičit", jako je to u jiných metod filtrování a spam filtry mohou hned po nasazení úspěšně odchytávat spamy. Pravidla jsou implementována lidmi a můžou být velice obsáhlá. Nevýhodou naopak je to, že filtr potřebuje časté aktualizace pravidel. Jak jednou spammer příjde na to, jak filtr obejít, bez aktualizace pravidel už se spamu nezbavíme.

Příkladem těchto pravidel může být například testování datumu odeslání, kdy spamové e-maily často mívají datum odeslání z budoucnosti. Odestuje se tedy, jestli čas není z budoucnosti, protože je nesmysl, aby byl legitimní e-mail odeslán z budoucnosti.

Dalším příkladem pravidla může být regulární výraz: *v.0,2i.0,2a.0,2g.0,2r.0,2a* Tento regulární výraz hledá jedno z nejčastějších spamových slov, a to je slovo viagra. Spameři používají mnoho způsobů jak toto slovo napsat tak, aby bylo pro člověka čitelné a přitom zůstalo skryto před spam filtry. Pomocí tohoto regulárního výrazu jej můžeme odhalit i ve tvarech viiaagra apod.

2.5 Statistické filtrování

V roce 1998 byla objevena metoda, která ukázala, že je možno filtrovat nevyžádanou poštu na základě statistického ohodnocení. Od té doby se používá statistické filtrování. Důvod pro tuto metodu je jednoduchý: jsou jednoduché na implementaci, mají velice dobré výsledky a vyžadují malou údržbu. Statistické filtry vyžadují „trénink" v podobě čtení nevyžádané i vyžádané pošty, poté, co se naučí rozeznat nevyžádané a vyžádané zprávy, se stává více efektivní. Jsou učeny uživateli, kteří označí příchozí e-mail za nevyžádanou nebo vyžádanou poštu, v budoucnosti se už podle toho filtr rozhodne sám. Tyto filtry využívají Bayesova algoritmu[5] pro statistické ohodnocení.

[4] velice oblíbený open-source spam filtr pro Apache servery

[5] V teorii pravděpodobnosti nazýváme Bayesovou větou vztah mezi pravděpodobností $P(E|F)$ a pravděpodobností opačně podmíněného jevu.

3 Fuzzy spam filtr

3.1 Úvod

Tento spam filtr využívá 5 základních pravidel, které ohodnotí přijatý e-mail a označí jej následně jako spam nebo ham. Při rozhodování využívá blacklistů i whitelistů, takže je na uživateli, aby tyto seznamy aktualizoval. Právě pomocí seznamů slov si může uživatel filtr přizpůsobit.

3.2 Pravidla

Zde je seznam a popis jednotlivých pravidel, která Fuzzy Spam Filtr používá. Každé pravidlo má svůj tzv. AttackFactor, což je označení, jak moc je e-mail podobný spamu. O rozhodování na základě AttackFactoru se dozvíme v následujících kapitolách.

Na rozdíl od originální implementace, která je popsána v originání dokumentaci filtru [1], vynecháme pravidlo č. 2, protože kolekce testovaných e-mailů neobsahuje IP adresy. Zde je uvedeno jen proto, aby byla zmíněna všechna možná pravidla Fuzzy Spam filtru, dále se tímto pravidlem zabývat nebudeme.

3.2.1 Pravidlo 1

a) $Pokud\ \exists\ AdresaOdesilatele \in EmailBlacklist \Rightarrow AttackFactor = -0,25$

b) $Pokud\ \exists\ AdresaOdesilatele \in EmailWhitelist \Rightarrow AttackFactor = 0,25$

c) $Pokud\ \exists\ AdresaOdesilatele \notin EmailBlacklist \& \exists AdresaOdesilatele \notin EmailWhitelist \Rightarrow AttackFactor = 0$

Vysvětlení

Pravidlo 1.a: Pokud existuje adresa odesílatele, která je ve EmailBlacklist, pak nastav AttackFactor tohoto pravidla na -0,25.

Pravidlo 1.b: Pokud existuje adresa odesílatele, která je ve EmailWhitelist, pak nastav AttackFactor tohoto pravidla na 0,25.

Pravidlo 1.c: Pokud existuje adresa odesílatele, která není ve EmailBlacklist ani v EmailWhitelist, pak nastav Attack factor tohoto pravidla na 0.

3.2.2 Pravidlo 2

a) $Pokud\ \exists\ IP\_odesilatele \in IPBlacklist \Rightarrow AttackFactor = -0,25$

b) $Pokud\ \exists\ IP\_odesilatele \in IPWhitelist \Rightarrow AttackFactor = 0,25$

c) $Pokud\ \exists\ IP\_odesilatele \notin IPBlacklist\ \&\ \exists\ IP\_odesilatele \notin IPWhitelist \Rightarrow AttackFactor = 0$

Vysvětlení

Pravidlo 2.a: Pokud existuje IP adresa odesílatele, která je v IPBlacklist, pak nastav AttackFactor tohoto pravidla na -0,25.

Pravidlo 2.b: Pokud existuje IP adresa odesílatele, která je v IPWhitelist, pak nastav AttackFactor tohoto pravidla na 0,25.

Pravidlo 2.c: Pokud existuje IP adresa odesílatele, která není v IPBlacklist ani v IPWhitelist, pak nastav AttackFactor tohoto pravidla na 0.

3.2.3 Pravidlo 3

a) $Pokud\ \forall\ Predmet \in WordsBlacklist \Rightarrow AttackFactor = -0,50$

b) $Pokud\ \exists\ Predmet \in WordsWhitelist \Rightarrow -0,50 < AttackFactor < 0,50$

Vysvětlení

Pravidlo 3.a: Pokud všechna slova obsažena v předmětu jsou mezi nevyžádanými slovy v Blacklistu, pak nastav AttackFactor na -0,50.

Pravidlo 3.b: Pokud existuje alespoň jedno slovo v předmětu, které je obsaženo mezi nevyžádanými slovy v Blacklistu, pak nastav AttackFactor mezi -0,50 a 0,50.

3.2.4 Pravidlo 4

a) $Pokud \ \forall \ Obsah \in WordsBlacklist \Rightarrow AttackFactor = -0,50$

b) $Pokud \ \exists \ Obsah \in WordsWhitelist \Rightarrow -0,50 < AttackFactor < 0,50$

Vysvětlení

Pravidlo 4.a: Pokud všechna slova obsažena v obsahu e-mailu jsou mezi nevyžádanými slovy v Blacklistu, pak nastav AttackFactor na -0,50.

Pravidlo 4.b: Pokud existuje alespoň jedno slovo v obsahu e-mailu, které je obsaženo mezi nevyžádanými slovy v Blacklistu, pak nastav AttackFactor mezi -0,50 a 0,50.

3.2.5 Pravidlo 5

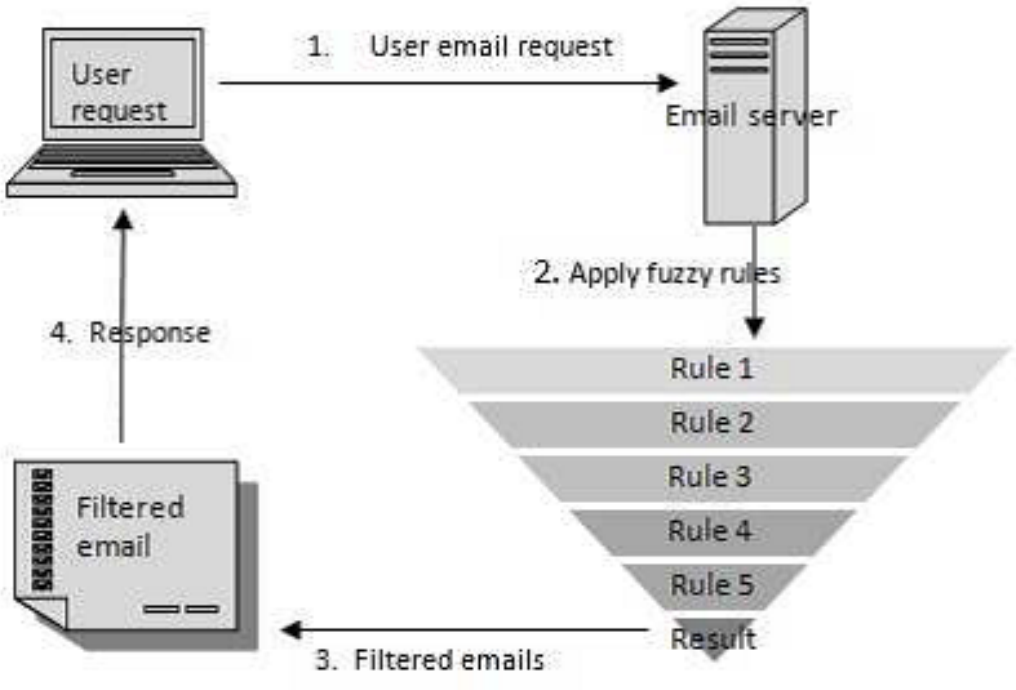
a) $Pokud \ \forall \ Prilohy \notin VirusList \Rightarrow AttackFactor = 1$

b) $Pokud \ \exists \ Priloha \in VirusList \Rightarrow AttackFactor = 1$

Vysvětlení

Pravidlo 5.a: Pokud příloha není obsažena ve VirusListu, pak nastav AttackFactor pro toto pravidlo na 1.

Pravidlo 5.b: Pokud existuje příloha, která je obsažena ve VirusListu, pak nastav AttackFactor pro toto pravidlo na 1.



Obrázek 1: Architektura Fuzzy navrženého systému

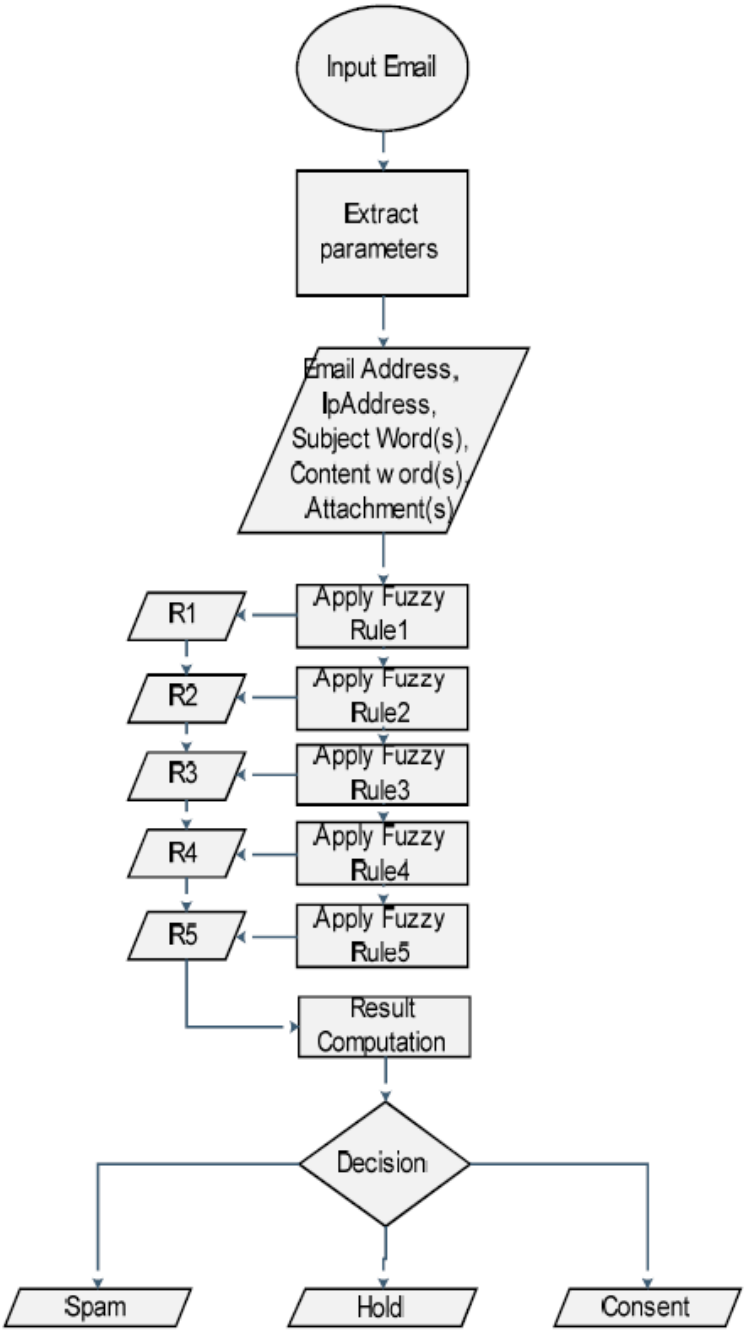
3.3 Implementace

Poté, co obdržíme e-mail, je odeslán do Fuzzy systému pro identifikaci, jak je vidět na obrázku 1[1]. Systém e-mail následně zkontroluje a odešle jej uživateli s případným označením o spamu. Detailní popis algoritmu vyhodnocení je zobrazen na obrázku 2[1].

V našem případě e-maily budeme číst ze složky, proto je poté ani nikam neodesíláme, jen e-mail dočasně označíme jako spam/ham. Toto označení proběhne pouze v paměti, kolekce se tedy nezmění.

Z popisu algoritmu si můžeme všimnout, že se vykonává pořád stejná práce dokola, jen ve velkém množství. Proto si do naší implementace přidáme rozdělění do vláken, aby následné testování probíhalo ryhleji. V praxi by pak stačilo pouze jedno vlákno.

Pravidlo 1 se vztahuje na adresu odesílatele. Nejdříve se vyextrahuje e-mailová adresa odesílatele ze zprávy, poté se pošle na kontrolu. Při kontrole se porovnává, jestli daná e-mailová adresa není obsažena v Blacklistu e-mailových adres (tedy nevyžádaných adres). Pokud se najde shoda, nastaví se Attack Factor pro toto pravidlo na -0,25. Poté se ta samá kontrola provede pro Whitelist (seznam vyžádaných adres), pokud se najde shoda tady, nastaví se AttackFactor na 0,25. Pokud se žádná shoda nenajde, AttackFactor se nastaví na 0.



Obrázek 2: Detailní popis Fuzzy algoritmu

Pravidlo 3 se vztahuje na předmět přijatého e-mailu. Předmět e-mailu může obsahovat jedno nebo více slov. Všechny tyto slova se postupně zkontrolují porovnáním s blacklistem nevyžádaných slov pro předmět. V originální dokumetaci můžeme zjistit, že se AttackFactor pro toto pravidlo počítá na základě četnosti slova v předmětu. Nám stačí najít jedno slovo z blacklistu a můžeme pak nastavit AttackFactor na -0,25, popř na 0,25, pokud je slovo obsaženo ve whitelistu. Tohle zjednodušení si můžeme dovolit, protože na rozdíl od originální implementace máme speciální blacklist a whitelist pro předmět e-mailu, jak bude uvedeno v následujíci kapitole 3.4.

Pravidlo 4 se vztahuje na obsah zprávy. Podobně jako v předchozím bodě rozdělíme tělo zprávy na jednotilvá slova. Slova pak porovnáme s blacklistem a whitelistem. AttackFactor se vypočítá podle následujícího algoritmu:

a) Rozdělíme obsah e-mailu na jednotlivé věty, řekněme W_i kde $i \geq 1$

b) Sečteme počet vět v e-mailu a označíme jej T_w

c) Pokud $T_w > 0$, pak pokračujeme krokem d).

d) Vypočítáme účinkový factor W_f kde $W_f = 0,5/T_w$

e) Provedeme porovnání pro každou větu W_i v Black listu

f) Pokud je nalezena shoda, potom nastavíme $W_{fi} = -W_f$ jinak $W_{fi} = W_f$; kde $i \geq T_w$

g) Vypočítáme $AttackFactor = \sum W_{fi}$

Pravidlo 5 se vztahuje na přílohy. Kontroluje se každá příloha zvlášť a to porovnáním se seznamem známých škodlivých příloh, které by mohly obsahovat např. virus. Pokud e-mail přílohu neobsahuje, pak AttackFactor pro toto pravidlo nastavíme na 0. Pokud je příloha nalezena v seznamu škodlivých příloh, tak nastavíme AttackFactor na -1, pokud není v tomto seznamu, pak nastavíme AttackFactor na 1.

3.4 Tvoření blacklistu a whitelistu

Jak již bylo zmíněno, spam filtr je silně závislý na vlastních blacklistech a whitelistech, které si tvoří uživatel sám. Proto potřebujeme nasimulovat učení spam filtru, abychom vytvořili vhodné seznamy. Filtr budeme učit velice jednoduchým, ač účinným způsobem. Všechny zprávy rozdělíme na slova a tyto slova si následně uložíme do seznamu slov, poté vybereme 500 nejpoužívanějších slov. Tím nám vzniknou naše seznamy - pro blacklist použijeme 500 nejčastějších slov z kolekce spamových e-mailů, pro whitelist použijeme 500 nejčastějších slov z kolekce hamových e-mailů. Tyto kolekce jsou poměrně rozsáhlé, protože na prvních místech v četnosti slov jsou slova normálně používaná a tak se tato slova v blacklistu a whitelistu navzájem ruší. Některé spam filtry používají k vyhodnocování také četnosti jednotlivých slov, aby vyhodnotili rizikovost slova, v našem případě tyto četnosti nepoužijeme, protože si je Fuzzy Spam filtr počítá sám, viz Pravidlo 4. Tímto

algoritmem tedy vytvoříme blacklisty a whitelisty pro slova v těle e-mailu, pro slova v předmětu a pro e-mailové adresy. Výsledný stav bude tedy následující:

- Počet slov v blacklistu pro tělo e-mailu: 500
- Počet slov v blacklistu pro předmět: 500
- Počet e-mailových adres v blacklistu: 100
- Počet slov v whitelistu pro tělo e-mailu: 500
- Počet slov v whitelistu pro předmět: 500
- Počet e-mailových adres v whitelistu: 100

Poznámka 3.1 Počet e-mailových adres je nižší, protože se mezi blacklisty a whitelisty neshodují, čili se neruši.

Pro seznam škodlivých příloh nám stači zkontrolovat, zda e-mail obsahuje přílohu. Pokud je e-mail ze spamové kolekce, jeho příloha je automaticky označena jako škodlivá. Toto si můžeme dovolit kvůli nízkému počtu příloh ve spamové kolekci, pouze 73 e-mailů z kolekce obsahuje přilohu, kolekce obsahuje celkem 24 910 e-mailů.

3.5 Přístup uživatele

Po získání všech předchozích hodnot musíme taky brát ohled na přístup uživatele. Přístup uživatele se nastavuje při implementaci. Přístup uživatele dělíme do tří skupin: High Positive, zero, high negative.
High positive – uživatelé v této skupině striktně odmítají spam
Zero – uživatelé v této skupině zůstávají neutrální a nemají žádná další omezení
High negative – uživatelům v této skupině občas nevadí přijmout spamový e-mail
Hodnoty pro vyhodnocení jsou následně určeny takto:
High positive $\geq$ 0,25;
Zero = 0;
High negative $\leq$ -0,25;
Rozhodování pro uživatele ve skupině High Positive:

- Pokud je nastaven přístup uživatele na high positive a attack factor všech pravidel je $>$ 0,25, pak je e-mail označen jako legitimní
- Pokud je nastaven přístup uživatele na high positive a attack factor jakéhokoliv pravidla je mezi 0,25 a 0, pak je e-mail označen jako pozdržený
- Pokud je nastaven přístup uživatele na high positive a attack factor jakéhokoliv pravidla je $<$ 0, pak je e-mail označen jako spam

Rozhodování pro uživatele ve skupině Zero

- Pokud je nastaven přístup uživatele na zero a attack factor všech pravidel je ≥ 0, pak je e-mail označen jako legitimní
- Pokud je nastaven přístup uživatele na zero a attack factor jakéhokoliv pravidla je < 0, pak je e-mail označen jako spam

Rozhodování pro uživatele ve skupině High Negative

- Pokud je nastaven přístup uživatele na High Negative a attack factor všech pravidel je $\geq$ -0,25, pak je e-mail označen jako legitimní
- Pokud je nastaven přístup uživatele na High Negative a attack factor jakéhokoliv pravidla je $<$ -0,25, pak je e-mail označen jako pozdržen, čili o výsledném vyhodnocení rozhodne uživatel

Pro naše účely nejlépe poslouží neutrální uživatel, protože filtr učíme strojově. V následujících kapitolách budeme tedy testovat za uživatele typu *zero*.

4 RuBaSt

4.1 Úvod

Tento spam filtr je osobní spam filtr založený na pravidlech, od toho je taky odvozen jeho název „RuBaST – Rule-Based Spam Terminator“. Tento spam filtr slouží jako poslední vrstva mezi schránkou uživatele a normálním obecným spam filtrem jako například již zmiňovaným SpamAssassin.

RuBaST je osobní filtr, a tak obsahuje uživatelovy osobní pravidla. Například pokud SpamAssassin propustí e-mail ze zahraničního diskuzního fóra a my u nich nemáme žádný účet, můžeme přidat pravidlo, které bude e-mailům z tohoto fóra dávat vysoké skóre, aby byly tyto zprávy následně označeny za spam. Taková filtrace by ve filtru SpamAssassin být nemohla, protože existují uživatelé, kteří e-maily z tohoto serveru očekávají a chtějí.

4.2 Popis implementace

Implementace filtru RuBaST v této práci bude velice odlišná od originálního filtru RuBaSt, protože originální filtr nemá zveřejněná pravidla pro filtrování, aby se je nedozvěděli spameři a nemohli tak najít a zneužít případnou chybu. Základ ovšem bude stejný, a to implementace aplikace, která předpřipraví vstupy pro pravidla napsána v jazyce Prolog. V našem případě bude tato aplikace napsána v jazyce C#, jádro prologu použijeme SWI-Prolog[6], který s naší aplikací propojíme pomocí knihovny SwiPlCs.dll[7].

4.3 C# aplikace

Nejdříve si musíme přípravit e-maily pro zpracování. Vyextrahujeme všechna slova z e-mailů a převedeme je na malá písmena. Odstraníme všechny znaky, kromě a-z a 0-9. Dále abychom omezili výskyt častých anglických slov jako je *the, a, i, am, you atd...* použijeme List of English Stop Words[8], tedy list často používaných anglických slov. Tyto slova vyfiltrujeme z vyextrahovaných slov z e-mailů.

Abychom zaručili, že filtr bude znát správná spamová slova, tak jej tyto slova naučíme[9]. Pro simulaci učení spam filtru použijeme Bayesův algoritmus[3]:

$$Pr(S|W) = \frac{Pr(W|S) \times Pr(S)}{Pr(W|S) \times Pr(S) + Pr(W|H) \times Pr(H)} \quad (1)$$

[6] http://www.swi-prolog.org/
[7] http://www.swi-prolog.org/contrib/CSharp.html
[8] http://norm.al/2009/04/14/list-of-english-stop-words/
[9] Tahle část je simulace uživatele, který by filtr později učil sám a pravidla si přidával

Kde:

- $Pr(S|W)$ označuje pravděpodobnost, že je e-mail spam, s ohledem na to, že obsahuje testovane slovo
- $Pr(S)$ označuje celkovou pravděpodobnost, že e-mail je spam
- $Pr(W|S)$ označuje pravděpodobnost, že se testované slovo nachází ve spamu
- $Pr(H)$ označuje pravděpodobnost, že e-mail je ham
- $Pr(W|H)$ označuje pravděpodobnost, že se testované slovo nachází v hamu

Pravděpodobnosti $Pr(W|S)$ a $Pr(W|H)$ zjistíme analýzou kolekce e-mailů. Vyhledáme vždy všechny výskyty daného slova a tento počet výskytů následně vydělíme počtem e-mailů v kolekci. Pomocí $Pr(S)$ a $Pr(H)$ pak nastavujeme přístup ke kolekci. Pokud nastavíme $Pr(S)$ na hodnotu 1, znamená to, že očekáváme, že daný e-mail je spam. Můžeme pak brát 1 = 100%. Z toho nám tedy vyplývá, že součet $Pr(S)$ a $Pr(H)$ musí být roven 1. V našem případě budeme na 80% předpokládat, že e-mail je spam, nastavíme tedy $Pr(S) = 0.8$ a $Pr(H) = 0.2$ [3].

Nakonec výslednou pravděpodobnost $Pr(S|W)$ vynásobíme číslem 100, abychom dostali pravděpodobnost v procentech. Tímto způsobem tedy dostaneme seznam pravidel pro tělo e-mailu. Tato pravidla převedeme do jazyka prolog a uložíme.

Většina uživatelů má určitý okruh lidí, od kterých pravidelně přijímá e-maily a ví, že tyto e-maily určitě neobsahují spam. Abychom zaručili, že tyto e-maily nebudou označeny za spam, vytvoříme seznam povolených e-mailových adres. Přečteme si tedy kolekci hamových e-mailů a vyextrahujeme adresu odesílatele, tyto adresy si uložime do seznamu[10]. Vybereme 100 nejčastějších adres a vytvoříme z nich pravidla v prologu. Vznikne nám soubor, který bude obsahovat e-mailovou adresu a její váhu, abychom zaručili, že e-mail nebude označen za spam, tak všechny e-maily v tomto seznamu dostanou ohodnocení -100. Toto ohodnocení znamená, že e-mail na 100% není spam. Popsaný postup můžeme opět vidět na obrázku 3.

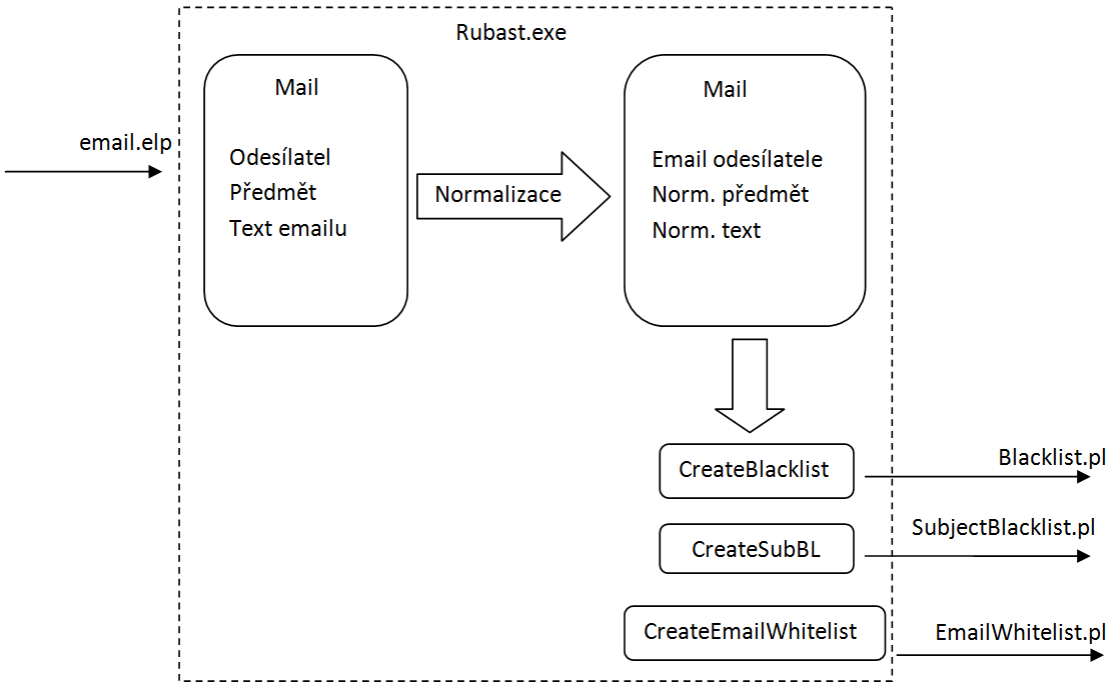
4.4 Prolog modul

Prolog modul představuje základ našeho spam filtru. Jsou v něm uloženy dva druhy filtrů:

- Words filter (dále jen WF), který se zabývá předmětem a tělem zprávy
- Rules filter (dále jen RF), který se zabývá obálkou e-mailu, tedy odesílatelem

Výsledné skóre pak vznikne kombinací těchto dvou filtrů.

[10]Opět se jedná o simulaci uživatele, konečný uživatel by si povolené e-mailové adresy přidával sám.



Obrázek 3: Architektura RuBaSt C# aplikace

4.4.1 Words filter

Spameři jsou často nuceni použít známa slova nebo výrazy, jako jsou třeba názvy léků, které chtějí prodat. Aby mohl WF tyto slova odhalit, připravili jsme si v C# aplikaci seznam pravidel, ve kterých tento modul vyhledává. Pravidlo vždy požaduje dva parametry - hledané slovo, proměnou, do které uloží jeho váhu. Vždy tedy vyhledáme požadované slovo a vrátíme jeho váhu, výsledné skóre se pak vypočítá následovně[4]:

$$words\_score = \sum_{i=1}^{|hitlist|} (Val[W_i] * Occ[W_i]) \tag{2}$$

Kde:

- $hitlist$ označuje testovanou kolekci
- $Val[W_i]$ označuje hodnotu spamového slova
- $Occ[W_i]$ označuje počet výskytů spamového slova

Stejný postup aplikujeme na předmět e-mailu. C# aplikace nám připravila pravidla, stačí nám tedy poslat jednotlivá slova do prolog jádra na otestování a dostaneme zpět výsledné ohodnocení, které se spočítá podle stejného vzorce jako je v rovnici 2.

4.4.2 Rules filter

Pokud příjmáme e-mail od známé e-amilové adresy, můžeme očekávat, že zpráva nebude spamová. Využijeme tedy připravených pravidel a pro každý e-mail odešleme adresu odesílatele na otestování do prolog jádra. Na otestování potřebujeme dva parametry: e-mailovou adresu, proměnou, do které uložíme výsledné ohodnocení. Volání takového otestování pak může vypadat následovně:

known_from('known_email_address', return_value).

Tímto způsobem otestujeme všechny e-maily v testované kolekci.

4.4.3 Přizpůsobení konečného skóre

Výsledné skóre je dáno počtem pravidel, které vrátily pozitivní výsledek, našly tedy shodu a vrátily spamové ohodnocení. Abychom omezili označení hamových e-mailů za spamové, započítáme do výsledku chybový faktor. Ten získáme tím, že si vypočítáme hustotu spamových slov[4]:

$$BWDensity = \frac{no\_of\_black\_words}{total\_words} \tag{3}$$

Kde:

- no_of_black_words označuje počet nalezených spamových slov
- total_words označuje celkový počet slov v testovaném e-mailu

Dále si pro přizpůsobení konečného skóre vypočítáme tzv. LinkScore, které počítáme na základě počtu nalezených odkazů, protože spamové e-maily velice často obsahují odkazy na jiné stránky, obvykle na stránky, kde se snaží prodat inzerovaný produkt.

$$LinkScore = 15 * \log_1 0(1 + \#links * (ruleScore + wordScore)) \tag{4}$$

Kde:

- LinkScore označuje výsledné skóre pro linky v e-mailu
- #links označuje počet linků v testovaném e-mailu
- ruleScore označuje hodnotu získanou z kapitoly 4.4.2
- wordScore označuje hodnotu získanou z kapitoly 4.4.1

4.5 Konečné skóre

Konečné skóre dostaneme kombinací všech předchozích výpočtů:

$$finalScore = linkScore + ruleScore + wordScore * BWDensity \tag{5}$$

Jako rozhodovací hranici pak nastavíme hodnotu 1. Dostaneme potom:

$$finalScore \geq 1 \Rightarrow email \in spam \tag{6}$$

$$finalScore < 1 \Rightarrow email \in ham \tag{7}$$

Z rovnice 6 vyplývá, že pokud konečné skóre přesáhne hodnotu 1, bude e-mail označen za spam. V opačném případě bude e-mail označen za ham.

5 Lazy approach for Spam detection

5.1 Úvod

Tento spam filtr se trochu liší od předchozích, a to hlavně tím, že obsahuje učící část přímo v originální implementaci[9]. Výsledkem učícího algoritmu je však opět seznam pravidel, pomocí kterých se snažíme odhalit spam. Není tedy problém, aby si uživatel připsal vlastní pravidla. Tento filtr se jako jeden z mála zaměřuje pouze na tělo zprávy, nezájímá ho tedy předmět ani odesílatel e-mailu.

5.2 Popis implementace

Nejprve si musíme připravit seznamy pravidel. Na to použijeme algoritmus, který spam filtr má. V tomto algoritmu se snažíme odhalit určité opakující se vzory, jak pro spamové e-maily, tak pro hamové. Místo vytvoření společných pravidel pro spamy i hamy vytvoříme rozdělené sety pravidel.

Pro vytvoření pravidel nejprve potřebujeme rozdělit e-mail na jednotlivá slova. Aby jsme se vyhnuli podobným slovům jako „Price“ a „price“, převedeme všechna slova na malá písmena. Obdobně jako u RuBaSt spam filtru použijeme také List of English stop words, zmiňovaný v kapitole 4.3. Když jsme si takto připravili vstup, můžeme začít jednotlivé vzory ohodnocovat. Ohodnocení probíhá podle následujícího vzorce[9]:

$$\sigma(X) = 100 * \frac{|\{M_j \in D | X \subseteq M_j\}|}{|D|} (kde 1 \leq j \leq |D|) \tag{8}$$

Kde:

- X je označení testovaného slova, tedy vzoru
- M_j je neprázdná množina slov, kde j je pořadové číslo zprávy
- D je konečná množina trénovacích e-mailů pro danou kategorii (tedy spam nebo ham)

Vzor je častý, pokud $\sigma(X) \geq \sigma_{min}$, kde σ_{min} je uživatelem specifikované minimum, kterého když vzor dosáhne, tak je označen za častý.

Po získání častých vzorů můžeme vytvořit tzv. Association rules - sdružená pravidla pro spam nebo ham. Takové pravidlo je pak ve formě $X \rightarrow c$, kde c je kategorie pro danou zprávu (spam / ham) a X je vzor. Z těchto pravidel musíme vytřídit silná pravidla, ohodnocení pravidla vypočítáme následovně[9]:

$$\theta(X \rightarrow c) = 100 * \frac{\sigma(X \cup c)}{\sigma(X)} \tag{9}$$

Pravidlo je pak silné, pokud $(X \rightarrow c) \geq \theta_{min}$, kde θ_{min} je uživatelem stanovené minimum, kterého když pravidlo dosáhne, je označeno za silné.

5.2.1 Tvorba k-vzorů

Pro celý algoritmus budeme potřebovat více vzorů. Tyto vzory se označují jako *k-vzory*, kde *k* znamená, kolik slov daný vzor obsahuje. Po vykonání předchozího algoritmu jsme tedy získali *1-vzory*. Následující vzory se vždy skládají ze vzorů předešlých. Pro tvorbu *2-vzorů* tedy použijeme *1-vzory*. Když si představíme množinu *1-vzorů* jako vektor, můžeme pak tento vektor vynásobit sám sebou a provést tak kartezský součin. Tím nám vznikne množina dvojic slov, které následně označíme jako *2-vzory*. Tímto způsobem pak můžeme pokračovat a vznikne nám algoritmus, kdy pro vytvoření vzoru *k-vzor* použijeme vzor *(k-1)-vzor*. V našem případě vytvoříme 3-vzory a dále pokračovat nebudeme.

5.3 Vyhodnocování

Po vykonání předchozích algoritmů máme k dispozici seznam silných pravidel pro spamové e-maily a pro hamové e-maily. Každý e-mail na začátku testování začíná s hodnotami $score(spam) = 1$ a $score(ham) = 1$ Následně e-maily rozdělíme na jednotlivé věty a porovnáváme je postupně se všemi pravidly. Při nalezení shody připočteme k patřičnému ohodnocení váhu pravidla (pro nalezení spamového pravidla připočteme do score(spam), při nalezení pravidla hamu připočteme do score(ham)). Následně můžeme vyhodnotit e-mail[9]:

$$Pokud\ \frac{score(spam)}{score(ham)}\ \geq\ \lambda\ pak\ email\ oznac\ za\ spam$$

Znak λ nám tedy představuje jakousi pomyslnou hranici, které rozděluje spam od hamu, tuto hranici si opět nastavuje uživatel sám. V našm případě nastavíme $\lambda = 1,5$. Tímto způsobem projdeme všechny e-maily v testované kolekci a označíme za spam nebo ham.

6 Testování

6.1 Testovací kolekce

Všechny spam filtry budeme testovat na jedné kolekci. Touto kolekcí bude *TREC 2006 Spam corpus* od NIST [11]. Tato kolekce e-mailů obsahuje *24 910 spamů* a *12 910 hamů*.

6.2 Testovací metoda

Jako testovací metodu jsem zvolil metodu testování Cross-validation [2], tedy křížovou validaci. Princip testování spočívá v tom, že rozdělíme testovanou kolekci na 10 částí, 9 částí využijeme pro simulované učení, poslední část pak pro samotné otestování. Výsledný model testování pak můžeme vidět v tabulce 1, kde vyznačený sloupec značí testovací část.

1	2	3	4	5	6	7	8	9	10
10	1	2	3	4	5	6	7	8	9
9	10	1	2	3	4	5	6	7	8
8	9	10	1	2	3	4	5	6	7
7	8	9	10	1	2	3	4	5	6
6	7	8	9	10	1	2	3	4	5
5	6	7	8	9	10	1	2	3	4
4	5	6	7	8	9	10	1	2	3
3	4	5	6	7	8	9	10	1	2
2	3	4	5	6	7	8	9	10	1

Tabulka 1: Cross-validaton

6.3 Test FuzzySpamFiltr

Rozdělíme testovací si kolekci na části podle kapitoly 6.3. Otestujeme každou část kolekce, kde pro každou část tvoříme nové blacklisty a whitelisty pomocí učící části. Výsledým testováním dostaneme 10 výsledků, které můžeme vidět v tabulce 2, kde FN zamená False negative - e-maily, které byly chybně označené jako hamové, FP znamená False positive - e-maily, které byly chybně označeny jako spamové. První sloupec pak označuje jednotlivé části z cross-validation, viz tabulka 1.

[11] http://trec.nist.gov/data/spam.html

Konečné hodnoty pro testování tedy budou:

- Počet e-mailů v celé ham kolekci: 12 910
- Počet e-mailů v celé spam kolekci: 24 910
- Počet e-mailů v jedné ham testovací části: 1 291
- Počet e-mailů v jedné spam testovací části: 2 491
- Počet částí: 10
- Počet e-mailových adres v blacklistu: 100
- Počet e-mailových adres v whitelistu: 100
- Počet slov v blacklistu předmětu: 500
- Počet slov v whitelistu předmětu: 500
- Počet slov v blacklistu pro tělo e-mailu: 500
- Počet slov v whitelistu pro tělo e-mailu: 500
- Počet příloh ve virus listu: max 73 - mění se se závislostí na aktuálně testovanou část
- Přístup uživatele *zero*

	Spam	Ham	FN	FP
1	81,03%	90,64%	18,97%	9,36%
2	81,77%	94,04%	18,23%	5,96%
3	81,29%	96,36%	18,71%	3,64%
4	81,74%	96,98%	18,26%	3,02%
5	83,58%	97,06%	16,42%	2,94%
6	85,27%	96,67%	14,73%	3,33%
7	85,75%	95,32%	14,25%	4,65%
8	84,50%	97,06%	15,50%	2,94%
9	85,07%	98,14%	14,93%	1,86%
10	82,86%	93,03%	17,14%	6,97%
Max	**85,75%**	**98,14%**	**18,97%**	**9,36%**
Min	**81,03%**	**90,64%**	**14,25%**	**1,86%**
Průměr	**83,29%**	**95,53%**	**16,71%**	**4,47%**

Tabulka 2: Výsledky testu Fuzzy spam filtru

6.3.1 Zhodnocení FuzzySpamFiltr

Z výsledků v tabulce 2 vidíme, že průměrně zachytíme *83,29%* spamu. Což není moc. Ale na druhou stranu propouštíme *95,53%* legitimních e-mailů, takže se nemusíme bát, že by námi chtěný e-mail byl označen za spam a následně nedorazil. Lepších výsledků by se dalo dosáhnout přizpusobením blacklistů nebo přidáním některých pravidel, například testováním IP adresy - jak je uvedeno v originální dokumentaci. Nebo testováním datumu odeslání, dále bychom mohli přiřadit ručně rizikovost určitým slovům, ale to vše už záleží na konečném uživateli. My jsme testovali pouze základní verzi spam filtru FuzzySpamFiltr.

6.4 Test RuBaSt

Testovací metodu zvolíme stejnou jako v případě FuzzySpamFiltru, vysvětlení metody naleznete v kapitole 6.3. Po vytvoření jednotlivých částí v cross-validation můžeme začít testovat, kdy se pokaždé vytvoří nové seznamy pravidel, podle učící části. Výsledky pak můžeme vidět v tabulce 3.

Kde:

- První sloupec značí testovanou část, viz. tabulka 1
- Spam označuje úspěšnost v odhalování spamů
- Ham označuje úspěšnost v odhalovaní hamů
- FN označuje „False negative“, tedy e-maily, které byly chybně označeny za hamové
- FP označuje „False possitive“, tedy e-maily, které byly chybně označeny jako spamové

6.4.1 Zhodnocení RuBaSt

Z naměřených výsledků v tabulce 3 vidíme, že průměrná úspěšnost v odhalovaní spamů byla *81,29%*, což není mnoho. Takových výsledků jsme dosáhli díky Bayesovu algoritmu, pomocí kterého jsme vytvořili seznam pravidel. Přizpůsobením pravidel by jsme zajisté dosáhli lepších výsledků. Tím by jsme ovšem mohli snížit úspěšnost v propuštěných hamových e-mailech, kde jsme průměrnou propustnost měli *96,11%* a v jednom případě dokonce *98,99%*, což je velice dobrý výsledek. K odhalování spamu jsme však ani zdaleka nevyužili plný potenciál filtru RuBaSt. V originální implementaci[5] jsou další, komplexnější pravidla a rozhodovací algoritmy, které naplno využívají možnosti jazyka Prolog. Bez znalosti těchto pravidel je však velmi obtížné tento filtr naimplementovat.

	Spam	Ham	FN	FP
1	79,61%	88,07%	20,39%	11,93%
2	80,89%	95,58%	19,11%	4,42%
3	81,33%	97,13%	18,67%	2,87%
4	81,89%	91,63%	18,11%	8,37%
5	79,77%	98,68%	20,23%	1,32%
6	81,49%	97,99%	18,51%	2,01%
7	80,89%	96,9%	19,11%	3,10%
8	81,57%	98,99%	18,43%	1,01%
9	83,34%	97,91%	16,66%	2,09%
10	82,10%	97,37%	17,90%	2,63%
Max	**83,34%**	**98,99%**	**20,39%**	**11,93%**
Min	**79,61%**	**88,07%**	**16,66%**	**1,01%**
Průměr	**81,29%**	**96,11%**	**18,71%**	**3,96%**

Tabulka 3: Výsledky testu spam filtru RuBaSt

6.5 Test LazyApproach

Pro testování byla opět použita testovací metoda Cross-validation, popsaná v kapitole 6.3. Po rozdělení testovací kolekce na části začneme testovat. Z každé učící části se vytvoří nové seznamy vzorů, které pak budou sloužit pro samotné testování. Výsledky testování můžeme vidět v tabulce 4.

6.5.1 Zhodnocení LazyApproach

Z výsledků vidíme, že filtr nebyl moc úspěšný, jak v odhalování spamů, tak v propouštění hamů. To může být zapříčiněno především tím, že filtr se zaměřuje pouze na obsah těla e-mailu. Tím se ochuzujeme o vytvoření seznamu povolených a zakázaných e-mailových adres, které pro odhalení spamu slouží nejlépe a často jsou nejspolehlivější. Pokud budeme brát v úvahu, že testujeme pouze tělo e-mailu, výsledky pak jsou docela vyhovující. Spamové e-maily totiž obsahují řádově 2x až 3x méně textu než hamové e-maily. Proto si vyzkoušíme k tomuto spam filtru pár testovacích metod přidat, aby jsme zlepšili jeho výsledky. Přidáme tvorbu 1-vzorů také z předmětů e-mailu a vybereme 100 nejčastějších e-mailových adres z hamových e-mailů a vytvoříme z nich seznam povolených adres. Výsledky tohoto testu pak můžeme vidět v tabulce 5. Z výsledků můžeme vidět malé zlepšení v odhalení spamu, naproti tomu zhoršení v propustnosti hamových e-mailů. Přizpůsobováním a vytvářením dalších seznamů pravidel by jsme mohli dosáhnout lepších výsledků, to už ale záleží na každém uživateli.

	Spam	Ham	FN	FP
1	74,11%	68,09%	25,89%	31,91%
2	73,87%	76,92%	26,13%	23,08%
3	75,79%	78,78%	24,21%	21,22%
4	89,96%	38,11%	10,04%	61,89%
5	70,49%	87,22%	29,51%	12,78%
6	74,19%	85,98%	25,81%	14,02%
7	73,87%	82,65%	26,13%	17,35%
8	72,42%	86,60%	27,58%	13,40%
9	75,75%	82,03%	24,25%	19,97%
10	74,63%	79,63%	25,37%	20,37%
Max	**89,96%**	**87,22%**	**29,51%**	**61,89%**
Min	**70,49%**	**38,11%**	**10,04%**	**12,78%**
Průměr	**75,51%**	**76,60%**	**24,49%**	**23,60%**

Tabulka 4: Výsledky testu 1 spam filtru Lazy approach

	Spam	Ham	FN	FP
1	83,10%	65,38%	16,90%	34,62%
2	81,85%	70,41%	18,15%	29,59%
3	85,87%	70,88%	14,13%	29,12%
4	95,42%	28,12%	4,58%	71,88%
5	80,89%	79,47%	19,11%	20,53%
6	83,34%	77,92%	16,66%	22,08%
7	83,22%	74,52%	16,78%	25,48%
8	82,30%	78,23%	17,70%	21,77%
9	84,02%	74,67%	15,98%	25,33%
10	83,06%	70,64%	16,94%	29,36%
Max	**95,42%**	**79,47%**	**19,11%**	**71,88%**
Min	**81,85%**	**28,12%**	**4,58%**	**20,53%**
Průměr	**84,31%**	**69,02%**	**15,69%**	**30,98%**

Tabulka 5: Výsledky testu 2 spam filtru Lazy approach

6.6 Shrnutí naměřených výsledků

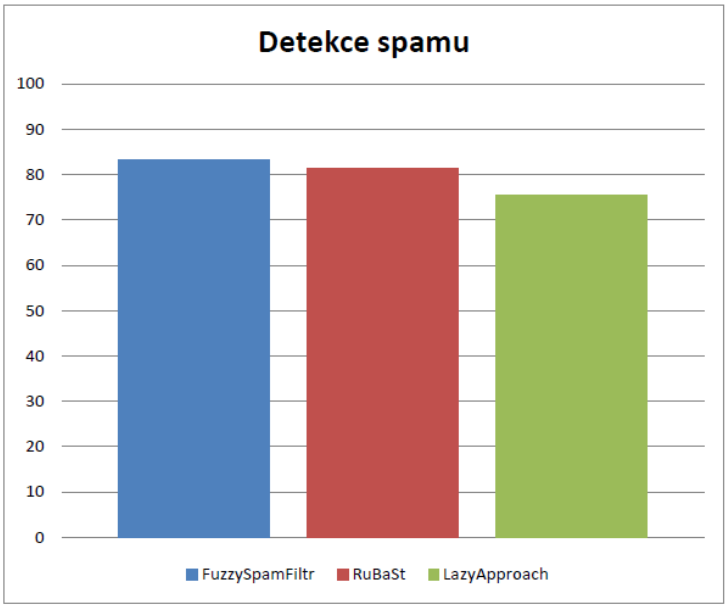
Všechny spam filtry jsou založeny na pravidlech, avšak každý má odlišný přístup. Nebo jsme se snažili odlišný přístup nasimulovat změnou učícího algoritmu. FuzzySpamFiltr je založen na oddělených seznamech pravidel, pro předmět, e-mailové adresy a tělo e-mailu. Seznamy pravidel jsou poměrně rozsáhlé oproti dvěma dalším filtrům a tomu také odpovídají výsledky. FuzzySpamFiltr dosáhl nejlepších výsledků, hned za ním se umístil RuBaSt.

RuBaSt na rozdíl od FuzzySpamFiltru má pouze jeden seznam pravidel, který jsme odlišili od fuzzyho pravidel použitím bayesova algoritmu. Tento spam filtr zdaleka neodpovídá originálu a je na něm asi nejvíc co vylepšovat, hlavně co se prolog jádra týče. V naší implementaci jsme použili prolog pouze pro uložení pravidel, nevyužili jsme jeho základních vlastností jako rekurzivního vykonávání podmínek atd.

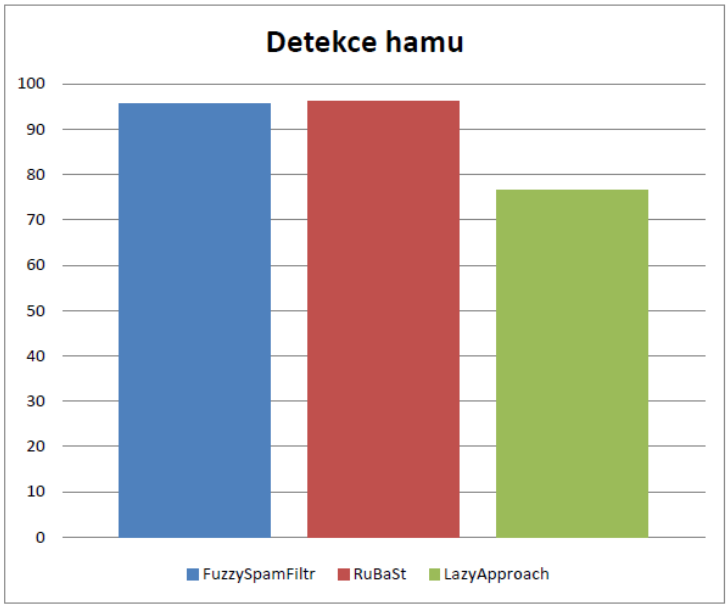
Nejhůře dopadl spam filtr LazyApproach, který používá pouze pravidla pro testování těla e-mailu. Ani po lehkém vylepšení filtru jsme se nedočkali dobrých výsledků, hlavně kvůli problémové části 4, která tomuto filtru dělá problémy, i když ostatní dva filtry u ní dosahují velice dobrých výsledků.

Porovnání výsledků si můžete prohlédnout na následujících grafech. Na obrázku 4 je graf, na němž je porovnání v úspěšnosti detekce spamů, úspěšnost je zde měřena v procentech a čím víc odchyceného spamu, tím líp. Na obrázku 5 je pak graf, který znázorňuje porovnání v úspěšnosti detekce hamu, stejně jako v předchozím případě, čím víc odhalíme hamů, tím líp.

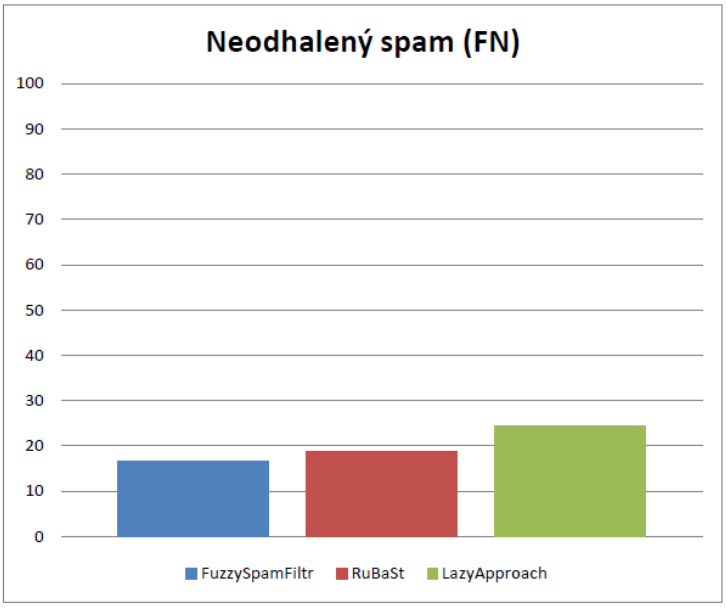
Naproti tomu na obrázcích 6 a 7 vidíme porovnání chybovosti, kde čím menší chybovost, tím líp.



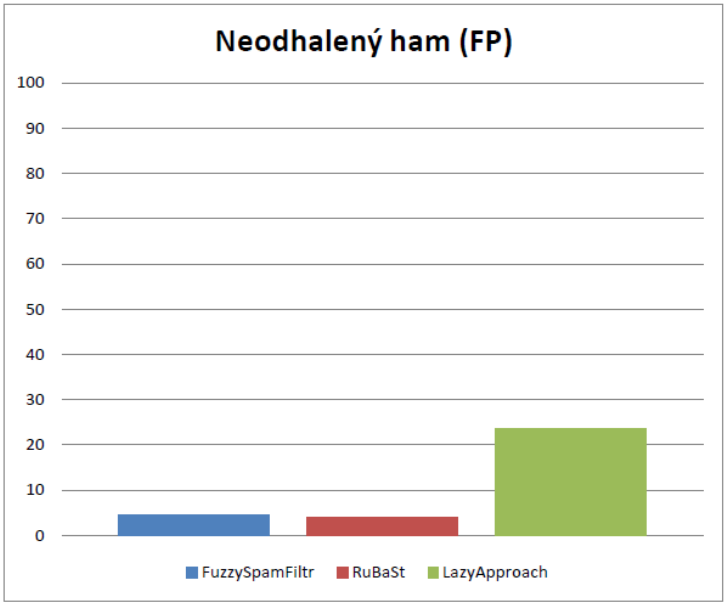
Obrázek 4: Porovnání výsledků v detekci spamu



Obrázek 5: Porovnání výsledků v detekci hamu



Obrázek 6: Porovnání výsledků v neodhaleném spamu



Obrázek 7: Porovnání výsledkl v neodhaleném hamu

7 Závěr

V této práci jsme si ověřili a naimplementovali funkčnost tří spam filtrů založených na pravidlech. Pro každý filtr jsme volili odlišný přístup. Žádný z námi naimpmlementovaných filtrů nedosahoval výsledků, které by byly pro ostrý provoz potřeba, tedy alespoň 90% úspěšnost v odchytávání spamu. V dnešní době se už spam filtry čistě založené na pravidlech moc nepoužívají, i když stále mají své místo. Slouží spíše jako podpora složitějších spam filtrů založených na bayesově algoritmu. Také se v dnešní době vyvíjejí spam filtry založené na neuronové síti[10], které se dokáží samy rozhodovat. Všechny tyto filtry však potřebují vstup od uživatele, aby rozlišil příchozí zprávy a filtry se tak mohly učit. Dá se říci, že na stejném principu fungují spam filtry Googlu a podobných velkých společností. Díky velkému počtu uživatelů maji obrovskou databázi spamových a hamových e-mailů a díky tomu dokáží vytvořit obecná pravidla pro všechny. Jednotliví uživatelé si pak mohou dodávat ostatní pravidla sami, zejména tím, že budou označovat nevyžádané e-maily jako spamové. Nakonec tedy vše záleží na tom, jak moc se uživatel o svůj spam filtr stará.

8 Reference

[1] P. Sudhakar, G. PoonKuzhali, K. Thaiagarajan, K. Sarukesi *Terminator for E-mail Spam - A Fuzzy Approach Revealed,* India, 2011.

[2] P.Refaeilzadeh, L. Tang, H. Liu *Cross-Validation,* Arizona State University, 2008.

[3] M. Prílepok, J. Platoš, V. Snášel, E. El-Qawasmeh *The Bayesian Spam Filter with NCD,* VSB - Technical University of Ostrava, 2012.

[4] G. Fiumara, M. Marchi, R. Pagano, A. Provetti, N. Spada *A rule-based system for end-user e-mail annotations,* University of Messina.

[5] G. Fiumara, M. Marchi, R. Pagano, A. Provetti *Rule-based Spam E-mail Annotation,* University of Messina.

[6] Jan Wielemaker *SWI Prolog Reference Manual,* University of Amsterdam, 2013

[7] R. Kamboj, V.P. Singh, S. Bhatia *A Rule Based Approach for spam detection,* Thapar University, 2010

[8] P. Olivka *Skripta k jazyku Prolog,* VSB - Technical University of Ostrava.

[9] A. Veloso, W. Meira Jr. *Lazy Associative Classification for Content-based Spam Detection,* Federal University of Minas Gerais, Brazil.

[10] J. Clark, I. Koprinska, J. Poon *A Neural Network Based Approach to Automated E-mail Classification,* School of Information Technologies, University of Sydney.

A Manuál k přiloženým programům

Spam filtry jsou napsány jako tři oddělené konzolové aplikace. Nemají žádnou formu UI, protože se jedná o programy, které většinou jen spustíme a běží na pozadí. Každá konzolová aplikace u sebe má knihovnu tříd, ve které je algoritmus spam filtru. Pro spuštění filtrů FuzzySpamFiltr a LazyApproach není zapotřebí žádných speciálních programů. U filtru RuBaSt je potřeba mít nainstalováno jádro Prolog a navést cestu k tomuto jádru v konfiguračním souboru. Každý spam filtr na konci testování ukládá výsledky do textového souboru, aby byly výsledky líp čitelné, generujeme kód v LaTeXu. a následně jej překládáme do PDF souboru. Z toho důvodu je nutné mít nainstalováno jádro pro kompilaci latexového kódu, z důvodu kompatibility doporučuji TeXLive 2012.

Spuštění testu Před samotným spuštěním testu je nutné mít vytvořeny složky s testovanými e-maily. Přiložené DVD obsahuje rozdělenou kolekci (je potřeba zkontrolovat cestu k této kolekci v konfiguračním souboru, protože každý uživatel může mít DVD disk pod jiným označením) . Pokud by si chtěl uživatel vytvořit svou kolekci, jsou pro to připraveny metody pro rozdělení kolekce. Stačí si otevřít konfigurační soubor a nastavit příslušné cesty podle komentářů. V konfiguračním souboru si také může uživatel změnit počet vláken (pouze v případě FuzzySpamFiltru, protože jako jediný běží ve více vláknech). V konfiguračním souboru si také uživatel může nastavit kam se mu budou ukládat výsledky, jak v txt, tak v pdf podobě. Všechny složky, na které se v konfiguračním souboru odkazujeme, musí existovat.